1998

The Phoenix Autonomous Underwater Vehicle

Brutzman, Don

Brutzman, Don, Healey, Tony, Marco, Dave and McGhee, Bob, "The Phoenix Autonomous Underwater Vehicle," chapter 13, AI-Based Mobile Robots, editors David Kortenkamp, Pete Bonasso and Robin Murphy, MIT/AAAI Press, Cambridge Massachusetts, 1998.

http://hdl.handle.net/10945/36490

The *Phoenix* Autonomous Underwater Vehicle

Don Brutzman, Tony Healey, Dave Marco and Bob McGhee
Center for Autonomous Underwater Vehicle Research
Code UW/Br, Naval Postgraduate School
Monterey California 93943-5000 USA
brutzman@nps.navy.mil

Abstract. The *Phoenix* autonomous underwater vehicle (AUV) is a robot for student research in shallow-water sensing and control (Figure 1). *Phoenix* is neutrally buoyant at 387 pounds (176 kg) with a hull length of 7.2 feet (2.2 m). Multiple propellers, thrusters, plane surfaces and sonars make this robot highly controllable. The underwater environment provides numerous difficulties for robot builders: submerged hydrodynamics characteristics are complex and coupled in six spatial degrees of freedom, sonar is problematic, visual ranges are short and power endurance is limited. Numerous *Phoenix* contributions include artificial intelligence (AI) implementations for multisensor underwater navigation and a working three-layer software architecture for control. Specifically we have implemented the execution, tactical and strategic levels of the Rational Behavior Model (RBM) robot architecture. These three layers correspond to hard-real-time reactive control, soft-real-time sensor-based interaction, and long-term planning respectively. Operational software functionality is patterned after jobs performed by crew members on naval ships. Results from simple missions are now available.

In general, a critical bottleneck exists in AUV design and development. It is tremendously difficult to observe, communicate with and test underwater robots because they operate in a remote and hazardous environment where physical dynamics and sensing modalities are counterintuitive. Simulation-based design using an underwater virtual world has been a crucial advantage permitting rapid development of disparate software and hardware modules. A second architecture for an underwater virtual world is also presented which can comprehensively model all necessary functional characteristics of the real world in real time. This virtual world is designed from the perspective of the robot, enabling realistic AUV evaluation and testing in the laboratory. 3D real-time graphics are our window into the virtual world, enabling multiple observers to visualize complex interactions.

Networking considerations are crucial within and outside the robot. A networked architecture enables multiple robot processes and multiple world components to operate collectively in real time. Networking also permits world-wide observation and collaboration with other scientists interested in either robot or virtual world. Repeated validation of simulation extensions through real-world testing remains essential. Details are provided on process coordination, reactive behaviors, navigation, real-time sonar classification, path replanning around detected obstacles, networking, sonar and hydrodynamics modeling, and distributable computer graphics rendering. Finally in-water experimental results are presented and evaluated.

Figure 1. *Phoenix* AUV testing in Moss Landing Harbor, California.

1 INTRODUCTION

This work describes software architectures for an autonomous underwater robot and for a corresponding underwater virtual world, emphasizing the importance of 3D real-time visualization in all aspects of the design process. Recent work using the *Phoenix* AUV is notable for the successful implementation and integration of numerous software modules within

multiple software layers. The three-layer software architecture used is the Rational Behavior Model (RBM), consisting of reactive real-time control (execution level), near-real-time sensor analysis and operation (tactical level), and long-term mission planning and mission control (strategic level) (Byrnes 96) (Marco 96b). In effect a higher robot software layer also exists: an off-line mission assistant that uses rule-based constraints and means-ends analysis to help human supervisors specify mission details, followed by automatic generation of strategic level source code. Results for simultaneous operation of the three onboard robot software layers (running an autogenerated mission) have been verified by virtual world rehearsal and in-water testing (Davis 96a, 96b).

Theoretical development stresses a scalable distributed network approach, interoperability between models, physics-based reproduction of real-world response, and compatibility with open systems standards. Multiple component models are networked to provide interactive real-time response for robot and human users. Logical network connectivity of physical interactions is provided using standard sockets and the IEEE standard Distributed Interactive Simulation (DIS) protocol (IEEE 95). Implementation of the underwater virtual world and autonomous robot are tested using the actual *Phoenix* AUV (Figure 2).

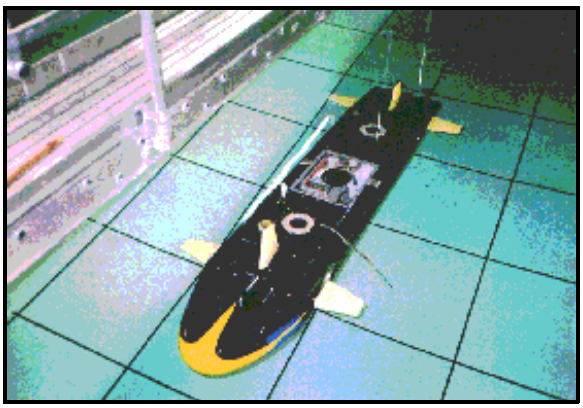


Figure 2. *Phoenix* AUV shown in test tank (Torsiello 94).

In order support repeatability of our results, documentation and source code are available electronically (Brutzman 96b, 96c). Current work includes model validation as well as adapting hydrodynamics and controls coefficients for other submersibles. Ongoing work also includes making 3D graphics and networking compatible with the Virtual Reality Modeling Language (VRML 2.0), to permit Internet-portable rendering and interaction via any computer connected to the World Wide Web.

Chapter Organization. Section 2 presents motivations for artificial intelligence (AI) approaches in underwater robotics. Section 3 describes robot hardware for *Phoenix*. Sections 4 through 7 examine the Rational Behavior Model (RBM) software architecture, detailing the execution, tactical and strategic levels. Section 8 describes robot networking. Sections 9 and 10 discuss virtual world design criteria and visualizing control algorithms. Section 11 presents AUV-virtual world communications which permit real-time physics-based response in the laboratory. Sections 12 and 13 discuss interactive 3D computer graphics and sonar visualization. Section 14 evaluates experimental results. Section 15 points out areas for future work. The chapter closes with conclusions, references, and pointers to a repository for software and documentation.

2 MOTIVATION

Untethered underwater robots are normally called Autonomous Underwater Vehicles (AUVs), not because they are intended to carry people but rather because they are designed to intelligently and independently convey sensors and payloads. AUVs must accomplish complex tasks and diverse missions, all while maintaining stable physical control with six spatial degrees of freedom (i.e. posture, meaning 3D position plus 3D orientation).

The underwater environment is highly challenging. Hydrodynamics forces are surprisingly cross-coupled between various axes because of asymmetric vehicle geometry and the nonlinear drag "added mass" of water fluid carried along with moving vehicles. Active sonar returns provide precise range but poor bearing accuracy, and can be subject to frequent dropouts. Sonar range maxima are highly frequency-dependent. At moderate ranges (beyond several hundred meters) sonar paths can bend significantly due to continuous refraction from sound speed variation, which is caused by changes in water temperature, salinity and pressure (i.e. depth). Vision is only possible for short ranges (tens of meters at best) and is often obscured if water is turbid. Underwater vision also requires powerful lighting, which is an unacceptable power drain due to already-severe power and propulsion endurance constraints. Laser sensors are usable to approximately 100 m range and provide good range and bearing data, but remain expensive, hard to tune and subject to turbidity interference. Typically little or no communication with distant human supervisors is possible. When compared to indoor, ground, airborne or space environments, the underwater domain typically imposes the most restrictive physical control and sensor limitations upon a robot. Underwater robot

considerations remain pertinent as worst-case examples relative to other environments (Figure 3).

- **Complex Hydrodynamics**
  - coupled in six spatial degrees of freedom
  - accompanying "added mass" of water
  - instability can be severe or fatal
- **Sonar**
  - accurate ranges but bearings poor
  - numerous nonlinear factors affect reverberation and attenuation
  - sonar path bending at long ranges due to sound speed profile (SSP) effects
- **Vision and Laser**
  - range limited by turbidity
  - lighting requires excessive power
- **Endurance typically a few hours**
  - limited power available
  - constrains all other equipment
- **Navigation**
  - ocean currents vary with time, location
  - acoustic navigation requires calibrated prepositioned transponder field
  - GPS and inertial methods possible
- **Communications**
  - tether is an unacceptable encumbrance
  - acoustic limited in bandwidth, range
  - optical extremely limited range

Figure 3. Environmental constraints for underwater robots are severe.

A large gap exists between the projections of theory and the actual practice of underwater robot design. Despite numerous remotely operated vehicles (ROVs) and a rich field of autonomous robot research results, few complete AUVs exist and their capabilities are limited. Cost, inaccessibility and scope of AUV design restrict the number and reach of players involved. Interactions and interdependencies between hardware and software problems are poorly understood. Equipment reliability and underwater electrical connections are constantly challenging. Testing is difficult, tedious, infrequent and potentially hazardous. Meaningful evaluation of results is hampered by overall problem complexity, sensor inadequacies and human inability to directly observe the robot *in situ*. Potential loss of an autonomous underwater robot is considered intolerable due to tremendous investments in time and resources, the likelihood that any failure will become catastrophic, and difficulty of underwater recovery.

Underwater robot progress is slow and painstaking for other reasons as well. By necessity most research is performed piecemeal and incrementally. For example, a narrow problem might be identified as suitable for solution by a particular AI paradigm and then examined in great detail. Conjectures and theories are used to create an implementation which is tested by building a model or simulation specifically suited to the problem in question. Test success or failure is used to interpret validity of conclusions. Unfortunately, integration of the design process or even final results into a working robot is often difficult or impossible. Lack of integrated testing prevents complete verification of conclusions.

AUV design must provide autonomy, stability and reliability with little tolerance for error. Control systems require particular attention since closed-form solutions for many hydrodynamics control problems are unknown. AI methodologies are thus essential for numerous critical robot software components. Historically, the interaction complexity and emergent behavior of multiple interacting AI processes has been poorly understood, incompletely tested and difficult to formally specify (Shank 91). We are happy to report that these problems can be overcome. Our three-layer robot software architecture, in combination with a physically and temporally realistic virtual world, has enabled effective research, design and implementation of an autonomous underwater robot.

The charter of the Naval Postgraduate School (NPS) Center for AUV Research group is to support graduate student thesis research. Certainly there is no shortage of problems that underwater robotics researchers might work on. We believe that having a clear and compelling objective is fundamentally important. Mission drives design. A well-defined goal provides priorities that can be understood by a large research group, clear criteria for making difficult design tradeoffs, and a finish line: success metrics are defined. We have chosen shallow-water minefield mapping as our driving application. At the 1995 Symposium on Autonomous Vehicles for Mine Countermeasures (MCM) (Bottoms 95), consensus was reached that all technical components exist which are needed to build effective MCM AUVs. Our motivating goal is to demonstrate such a vehicle. We intend to demonstrate that there are no fundamental technical impediments to mapping shallow-water minefields using affordable underwater robots. We are integrating component technologies necessary for underwater autonomy in a working system, and are making good progress toward reaching that goal.

Related efforts. Over a dozen other research groups are active in underwater robotics. The Massachusetts Institute of Technology (MIT) Sea Grant has deployed several *Odyssey*-class AUVs notable for open-ocean and under-ice oceanographic

exploration leading to the possibility of autonomous oceanographic sampling networks (AOSNs) (Curtin 93). The Florida Atlantic University (FAU) ocean engineering department has built a series of vehicles which include fuzzy logic controllers and special sensing techniques (Smith 94). The Woods Hole Oceanographic Institute (WHOI) Deep Submergence Lab (DSL) has specialized in long-term bottom monitoring, acoustic communications and remotely teleoperated task-level supervision of manipulators (Sayers 96). An excellent introductory text on underwater robot design and control is (Yuh 95). Annual AUV technical symposia are sponsored in alternate years by the IEEE Oceanic Engineering Society (OES) (*http://auvibm1.tamu.edu/oes*) and the Autonomous Undersea Systems Institute (AUSI) (*http://www.cdps.maine.edu/AUSI*).

Important problem domain for AI. Despite many handicaps, the numerous challenges of operating in the underwater environment force designers to build robots that are truly robust, autonomous, mobile and stable. This fits well with a motivating philosophy of Hans Moravec:

> .. solving the day to day problems of developing a mobile organism steers one in the direction of general intelligence... Mobile robotics may or may not be the fastest way to arrive at general human competence in machines, but I believe it is one of the surest roads. (Moravec 83)

3 HARDWARE

Detailed knowledge regarding robot capabilities and requirements are necessary prerequisites for designing and implementing robot software. Overview descriptions of the *Phoenix* AUV and related research appear in (Brutzman, Compton 91). Both an external view and internal vehicle component arrangements are shown in Figures 4 and 5.

Designed for research, the *Phoenix* AUV has four paired plane surfaces (eight fins total) and bidirectional twin propellers. The hull is made of pressed and welded aluminum. The vehicle is ballasted to be neutrally buoyant at 387 lb (176 kg) with a hull length of 7.2 ft (2.2 m). Design depth is very shallow at 20 ft (6.1 m). Two pairs of sealed lead-acid gel batteries provides vehicle endurance of 90-120 minutes. Since battery electrical discharge produces hydrogen gas, hydrogen absorber pellets reduce the potential hazard of explosion. Twin propellers provide 5 pounds of force (lbf) (22.5 N) with resulting speeds up to 2 knots (~1 m/sec). A free-flooding (vented to water) fiberglass sonar dome supports two forward-looking sonar transducers, a downward-looking sonar altimeter, a water speed flow meter and a depth pressure cell. Five rotational gyros mounted internally are used to measure angles and rates for roll, pitch and yaw respectively. Small cross-body thruster tunnels were locally designed and built for the *Phoenix* AUV. An in-line bidirectional propeller inside each thruster can provide up to 2 lbf (8.9 N). Detailed schematics and specifications of all *Phoenix* AUV hardware components are presented in (Torsiello 94).

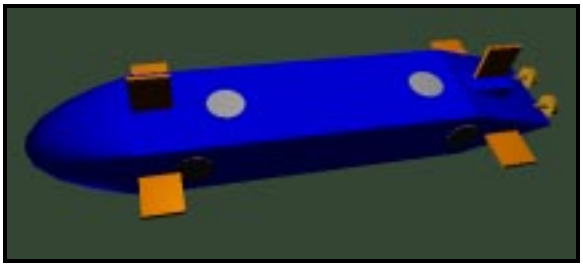


Figure 4. Exterior view of NPS *Phoenix* AUV.

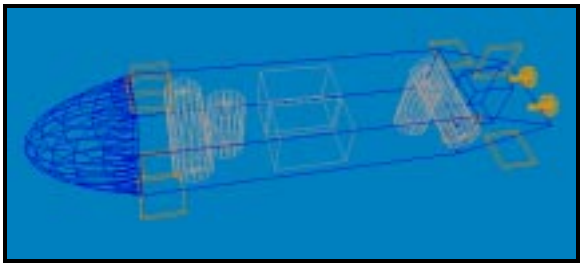


Figure 5. Internal view of NPS *Phoenix* AUV.

The primary computer for low-level hardware control is a GesPac 68030 running the OS-9 operating system. A significant recent hardware improvement was addition of a Sun Sparc 5 "Voyager" laptop workstation, with the display monitor removed to save space. Also connected is a paddlewheel speed sensor, depth sensor, DiveTracker acoustic navigation system (Flagg 94), Geographic Positioning System (GPS), Differential GPS (DGPS) and inertial navigation system (INS) equipment (Bachmann 96), as well as Ethernet local-area network (LAN) connections between onboard computers and (optionally) to external networks. Twin sonars have 1 cm resolution out to 30 m maximum range, with the ST725 (725 KHz) having a 1° wide by 24° vertical beam, and the ST1000 (1 MHz) a 1° conical beam. Each sonar is steered mechanically in 0.9° increments.

4 SOFTWARE OVERVIEW

The *Phoenix* AUV is primarily designed for research on autonomous dynamic control, sensing and AI. Software control of the vehicle is provided at a low level corresponding to maneuvering control of plane surfaces and propellers, as well as at a high level

corresponding to strategic planning and tactical coordination. Sensors are also controlled via execution level microprocessor-hardware interfaces, although some sensor functions may be optionally commanded by the intermediate tactical level, such as steering individual sonar transducer heading motors during classification.

Due to the large variety of critical tasks an autonomous underwater robot must perform, a robust multilevel software architecture is essential. Underwater robot software architectures are a particular challenge because they include a many of the hardest problems in robotics, control and AI over short, medium and long time scales.

Rational Behavior Model (RBM). The software architecture used by the *Phoenix* AUV is the Rational Behavior Model (RBM) (Byrnes 93, 96). The Rational Behavior Model (RBM) is a trilevel multiparadigm software architecture for the control of autonomous vehicles. Execution, tactical and strategic levels correspond roughly to direct interaction with vehicle hardware and environment, intermediate computational processing of symbolic goals, and high-level planning, respectively. The three levels of RBM correspond to levels of software abstraction which best match the functionality of associated tasks. Temporal requirements range from hard-real-time requirements at the execution level, where precise control of vehicle sensors and propulsion is necessary to prevent mission failure or vehicle damage, to soft-real-time long-term planning at the strategic level.

RBM provides an overall structure for the large variety of *Phoenix* AUV software components. A particular advantage of RBM is that the three levels of RBM can be informally compared to the watchstanding organization of a submarine crew (i.e. a manned AUV). Watchstanders operating vehicle sensors, the propulsion plant and diving station controls correspond to the execution level. Precise real-time control is needed at this level. The Officer Of the Deck (OOD) is represented in the tactical level, carrying out Commanding Officer (CO) orders by sending individual commands capable of being carried out by watchstanders at the execution level. Due to the diversity of tactical tasks and the complexity of some orders from the CO, the OOD has assistants at the tactical level to assist in their decomposition. These departments (navigation, sonar, path replanner etc.) permit the OOD to concentrate on sequencing and coordinating overall vehicle operation rather than exhaustively directing every detail. Finally the CO is responsible for mission generation and successful completion. CO tasks include mission-related planning and decision making, all performed at the strategic level. This architectural relationship is illustrated in Figure 6 (Holden 95).

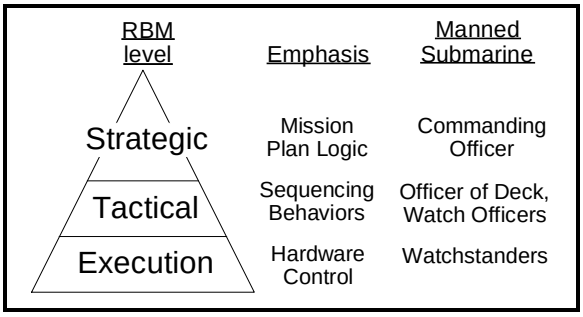


Figure 6. Rational Behavior Model (RBM) software architecture (Holden 95).

Human analogies are particularly useful for naval officers working on this project who already know how to drive ships, submarines and aircraft, since they provide a well-understood partitioning of duties and a clearly defined task lexicon. The naval analogies used here merely express common and essential robotics requirements using terminology familiar to the many officer students who have worked on *Phoenix*. This approach permits them to intuitively apply at-sea experience and domain knowledge. The RBM paradigm continues to serve well as a formal robot architecture which scalably composes numerous critical processes having dissimilar temporal and functional specifications.

RBM three levels summarized. Execution level software integration includes physical device control, sense-decide-act, reactive behaviors, connectivity, a mission script language, and stand-alone robustness in case of loss of higher levels. Tactical level software includes Officer of the Deck (OOD) coordination of parallel tactical processes, telemetry vector state variable updates as a form of shared memory, sonar control, sonar analysis and classification, path planning, DiveTracker acoustic navigation, DiveTracker acoustic communications, DGPS/GPS/INS navigation, and fail-safe mission abort if strategic level commands are lost. Strategic level software integration includes cross-language message passing, linking dissimilar binary executables, and several functionally equivalent strategic level variations: missions prescribed by Prolog rules, static mission scripts or an off-line mission generation expert system. There are numerous three-level robot architectures and many are similar to RBM.

Operating Systems and Compilers. Interestingly enough, operating system and compiler considerations have been most notable for their incompatibilities rather than their power. Aside from multitasking and

interprocess communications, we have not yet found it necessary (or desirable) to take advantage of real-time operating system constructs. The execution level resides on a GesPac 68030 under OS-9 written in Kernighan and Richie (K&R) C, a precursor to ANSI C. The tactical and strategic levels currently reside on the Voyager Sparc 5 laptop under Solaris Unix, written in ANSI C and Prolog respectively. Additionally, tactical and execution software can identically compile under SGI Irix 5.3 Unix in ANSI C. Compilation of single version source files across a variety operating system architectures and language variants is achieved through use of `#ifdef` and *Makefile* constructs (Brutzman 96c). This prevents "versionitis" or multiple file versions which inevitably lead to programmer confusion, incompatible source code interoperability and wasted effort. We are continuing this interoperability trend by porting to the well-supported public domain compiler *g++* (GNU ANSI C/C++).

Hierarchical versus reactive. Only a few years ago, robot architecture designers seemed preoccupied with bipolar arguments between hierarchical and reactive approaches. Hierarchical stereotypes included phrases like deliberative, symbolic, structured, "top down," goal-driven, explicit focus of attention, backward inferencing, world models, planning, search techniques, strictly defined goals, rigid, unresponsive in unpredicted situations, computation-intensive, and highly sophisticated performance. Reactive stereotypes included phrases like subsumptive, "bottom up," sensor-driven, layered, forward inferencing, robust subsuming behaviors, avoid both dynamic planning and world models, behave somewhat randomly, succeed without massive computations using well-considered behaviors, difficulty scaling up, elusive stability and nondeterministic performance. RBM is a hybrid architecture that is hierarchical at the top layer, reactive at the bottom layer and a mixture in between. Real-time responsiveness varies correspondingly at each level. From our experience with *Phoenix* it appears clear that a three-layer hybrid architecture is essential for a robot that must meet a broad range of timing requirements. Similar three-layer hybrid architectures now appear to be the norm for many mobile robots.

World models. Numerous *Phoenix* AUV theses and source code implementations have been handicapped by inadequate end-to-end hardware and software functionality within the vehicle. Such constraints are common for AUVs. Availability of networked hydrodynamics and sonar models for integrated simulation during robot development have been invaluable for development of robot control algorithms. This approach has permitted realistic development of software in all three software levels, independently and in concert, first in the virtual world and then in the real world.

Declaring that combined models create a virtual world rather than a simulation is not an overstatement. From the robots perspective, the virtual world can effectively duplicate the real world if robot hardware/software response is identical in each domain. In effect, this is a type of Turing test from the robot's perspective. Such a concept is controversial, perhaps especially among reactive behavior-based approaches which assume world models are unavoidably overcomplicated and use "the world is its own best model" (Brooks 86). In our case the challenges of the underwater environment eliminate relying on world availability throughout robot development. Development of a virtual world architecture that can realistically support the robot architecture has produced a new paradigm for robot software development (Brutzman 92a, 93, 94).

5 EXECUTION LEVEL

Disaster and divergence. In 1994 the execution level was the only software which effectively existed inside the *Phoenix* AUV. A second networked version of execution level was adapted to run in conjunction with developmental tactical routines and the underwater virtual world. A disastrous hydrogen explosion occurred in 1994 which required over a year to repair. During this reconstruction period many changes and enhancements were made to the AUV software. Unfortunately the two versions of execution level software grew far apart as they progressed, with the in-water version emphasizing new hardware interfaces (Healey, Marco 95) and the virtual world version emphasizing increased functionality (Brutzman 94).

Two versions into one. The top priority for 1995 efforts was to merge the two different versions of the execution level. The in-water code was painstakingly reintegrated with the virtual world version, one function at a time. This approach permitted frequent testing in the virtual world as well as continuous execution level accessibility to other tactical level work which proceeded in parallel. Laboratory bench tests were also conducted to ensure that software functions controlled the proper hardware and direction of rotation of moving components was correct. A single version of the combined execution level source code had to run on different computer architectures, using different compilers, and with different physical and logical interfaces. The new source code also had to run identically in the real world and the virtual world, all without error. This effort was successful (Burns 96) (Brutzman 96a).

Telemetry state vector. The execution level runs in a tight sense-decide-act loop and provides real-time control of vehicle sensors and effectors. Sensor data and effector orders are recorded in a telemetry state vector. This state vector is updated at the closed loop repetition rate, typically 6-10 Hz. The state vector is used for mission data recording, sharing critical parameters among all tactical processes, and providing a data-passing communications mechanism which permits identical operation in the real world and the virtual world (described later). State vector parameters, message-passing semantics and relation to flow of control are described in detail in (Brutzman 94).

Vehicle control. As current AUV research indicates, a great variety of control modes are possible when controlling vehicle posture and movement. A primary goal for the execution level is to provide robust open-loop and closed-loop control using propellers, cross-body thrusters and fin surfaces. Direct open-loop control of all these effectors is available, singly or in combination. Closed-loop control is available for course, depth and position, either in waypoint-follow mode or hover mode. Waypoint-follow mode relies on propellers and plane surfaces, which works well while transiting but poorly when stationary. Hover mode relies on propellers for short-range longitudinal motion, and thrusters for lateral/vertical/rotational motion. Hover mode allows precise station keeping in position, heading and depth, at least while dead-reckon position and ocean current set/drift estimates are accurate.

Mission script language. In keeping with our goal to make vehicle control understandable, we have implemented execution level functionality using a series of script commands. Each command consists of a keyword followed by a variable number of parameters. The mission script language controls operating modes and state flags in the execution level. A subset of the mission script language appears in Figure 7.

Commands can originate from tactical level processes, a prepared mission script file or a human operator. Each command is designed to be unambiguous and readable either by the robot or by people. Prescripted missions and tactical communications are intelligible because they sound similar to OOD orders and ship control party communications aboard ship. We believe this approach has general applicability for most AUVs. Another feature is text-to-speech conversion in the virtual world, simplifying human monitoring of mission progress. Overall execution level functionality also includes plotting telemetry results, replaying recorded mission telemetry data, and acting as network interface to sensor and hydrodynamics models when operating in the virtual world.

```
HELP                Provide keywords list

WAIT          #     Wait/run for # seconds

WAITUNTIL     #     Wait/run until clock time

QUIT                do not execute any more

RPM     # [##]      Prop ordered rpm values

COURSE        #     Set new ordered course

TURN          #     Change ordered course #

RUDDER        #     Force rudder to # degrees

DEPTH         #     Set new ordered depth

PLANES        #     Force planes to #

THRUSTERS-ON        Enable vertical and
                    lateral thruster control

NOTHRUSTER          Disable thruster control

ROTATE        #     open loop rotation
control

NOROTATE            disable open loop rotate

LATERAL       #     open loop lateral control


GPS-FIX             Proceed to shallow depth,
                          take GPS fix

GPS-FIX-COMPLETE    Surface GPS fix complete

GYRO-ERROR    #     Degrees of gyro error
                    [GYRO + ERROR = TRUE]

LOCATION-LAB        Vehicle is operating in
                    lab using virtual world.

LOCATION-WATER      Vehicle is operating in
                    water w/o virtual world.

POSITION  # ## [###]  reset dead reckon
                    i.e. navigation fix.

ORIENTATION  # ## ###  (phi, theta, psi)

POSTURE #a #b #c #d #e #f
              (x, y, z, phi, theta, psi)

OCEANCURRENT  #x #y [#z]

TRACE         verbose print statements

STANDOFF      #     Change standoff distance
                    for WAYPOINT-FOLLOW,
HOVER

WAYPOINT      #X #Y [#Z]

HOVER         [#X #Y] [#Z] [#orientation]
              [#standoff-distance]
```

Figure 7. Mission script language (from file *mission.script.HELP*) (Brutzman 94).

6 TACTICAL LEVEL

Officer of the Deck (OOD) Coordination. Of the three levels of the RBM architecture, the tactical level

was the last developed onboard *Phoenix*. Creation of an OOD module is crucial. The OOD controls the flow of information between other levels and within the tactical level, yet cannot become overburdened by unnecessary details. By forking parallel processes, the OOD creates several departments which are available to assist in processing commands and sensor data. Reuse of execution level functions and data structures reduces the amount of unique code needed by the tactical level. A modular interface design permitted the departments and OOD to be developed simultaneously. Figure 8 shows interprocess communications (IPC) from OOD to strategic level, execution level and other tactical level processes (Leonhardt 96).

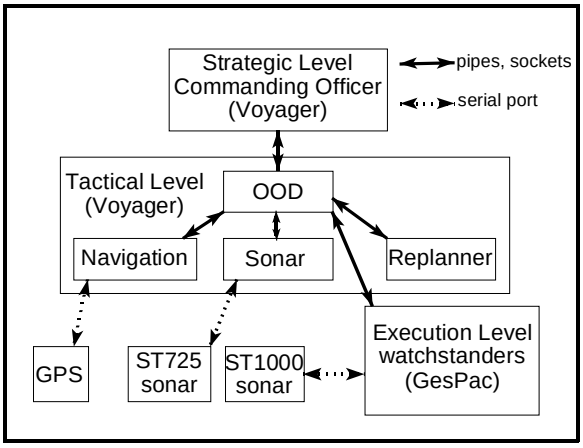


Figure 8. Interprocess communications (IPC) (Campbell 96).

Properly implementing IPC is crucial. Forked Unix processes have duplicate variable stores but do not share memory. Thus state variable changes in the parent (OOD) and children processes (navigation, sonar, replanner) must be performed individually for each process. We use standard Unix pipes for this communication since the tactical level is always within a single processor (Stevens 95). BSD-compliant sockets are used for communications to the execution level since that operates on a different processor (or even on a different network). Separate communication channels are used for updating state vectors and exchanging orders/ acknowledgements.

Navigation. The navigation module is a parallel forked process of the tactical level. It uses an Asynchronous Discrete Kalman Filter to filter GPS satellite navigation data received from a Motorola 8-channel GPS/DGPS unit and ranges received from a commercial short baseline sonar range system (DiveTracker).

The *Phoenix* is designed for precision navigation requiring position accuracy of 1 m. The standard deviation of the position available from GPS is approximately 60 m, with DGPS being accurate within 2 m. The DiveTracker short baseline acoustic ranges have a geometry-dependent standard deviation within 20 cm (with an occasional range out to 33 cm) which can cause a transiting position uncertainty of 1-3 m. Using raw positions results in fix-to-fix position uncertainty, control chattering and hydrodynamic stability problems for *Phoenix*. Kalman filtering corrects these difficulties.

Kalman filtering is a method for recursively updating an estimate of the state of a system by processing a succession of measurements. The *Phoenix* implementation uses a model-based movement estimator for state, combined with measurements, to produce the most probable estimate of the vehicle's position. A discrete Kalman filter is used to process measurements, and the use of acoustic range data requires an Extended Kalman Filter mode of operation due to the nonlinearity of range measurements (Bachmann 96).

Accurate and efficient navigation from point to point also requires the knowledge of the local ocean currents to prevent undershooting the intercept course towards the desired location. If a vehicle fix determines that the vehicle is not where the motion model predicts, then the likely causes are ocean current or AUV speed/heading errors. Using a non-zero mean movement model (where input vehicle speed is assumed truth) results in the filter solving for both an updated position data and estimates of ocean current. Estimated ocean currents are actually the combined sum of actual ocean current, errors in reported speed and heading errors. The ocean current values produced can thus change with the vehicle heading, but the root mean squared value of the currents will converge to a steady state number. This number can be resolved to X/Y or set/drift (polar) components for dead reckoning use. As with most processes at the tactical level, the algorithmic basis for this approach is similar to techniques used by human navigators.

By monitoring the difference between a motion model and measurements, the Kalman filter can determine if it has possibly lost track or received a bad measurement. If the difference is briefly too high, then the measurement is ignored. If the difference is too high for longer than 15 seconds, then it is assumed that the filter has lost track. Upon loss of track, the tactical level is informed and the OOD surfaces to gain a GPS fix and reset the filter state and parameters. This GPS-FIX procedure is designed to work equally well in hover and waypoint control. Full navigator details are in (McClarin 96) (Bachmann 96).

Real-time Sonar Classification. Real-time sonar classification and run-time collision avoidance are essential for AUV autonomy and survivability. An off-line sonar classification expert system was originally written using the CLIPS expert system shell (Brutzman 92b, 92c). Successful development of rules was originally dependent on the support of the expert system rule-matching engine. Once the expert system was developed, translation to C was practical and the optimized sonar classifier is now capable of running in real time to meet robot sensing requirements (Campbell 96).

The sonar module initializes sonar transducer parameters for maximum range scale, orientation change step size and transmitter power settings. Three modes are available: "transit search," "sonar search," and "rotate search." The transit search consists of a 60° sonar scan in front of the AUV. This search is primarily conducted for collision avoidance. The other two modes are conducted in a search area to detect, localize and classify any unknown objects. Sonar search and rotate search are 360° searches. Sonar search is performed by mechanically rotating the sonar head, whereas rotate search is accomplished with the sonar head fixed while the full *Phoenix* body performs a 360° rotation.

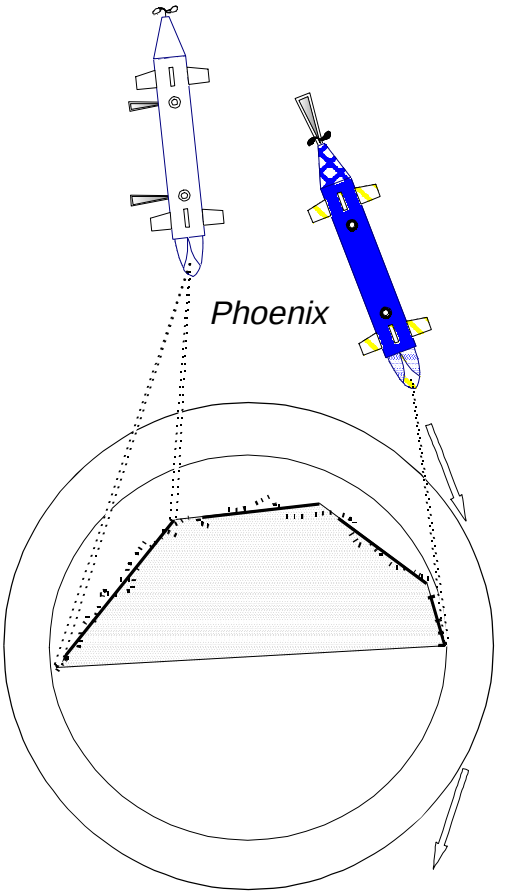
Sonar processing begins with filtering, thresholding and smoothing of the raw sonar data to produce a return bearing and range. The returns are then fitted to line segments using parametric regression. Line segments are started when a sliding window locates four returns that form an acceptable line. Points are subsequently added based on distance from the line segment and whether the new resultant line segment is acceptable. Completed line segments are then combined based on proximity and orientation.

To remove the directionality effects of sonar scan rotation, comparison of line segments is performed by first using the segment that is more clockwise relative to the AUV. Once objects and line segments are formed, heuristic rules are applied to classify the objects. The last part of the classification process is to relay object information in a manner suitable for path planning purposes. A circle representation is used with the center at the centroid of the object. Particularly long line segments (i.e. walls) are converted to a set of small adjacent circles. This methodology works. Additional experimental results are needed to ensure that system coefficients are properly tuned for current *Phoenix* sonars.

Imminent collision avoidance is achieved with a simple relative bearing and range check for all valid returns that contribute to any line segment. If a return does not contribute to a line segment it is not evaluated and is treated as a spurious return. We have developed more robust imminent collision avoidance algorithms independent of near-real-time sonar classification using the second steerable sonar. Using multiple noninterfering sonars permits employing search techniques that are otherwise mutually exclusive when sharing a single sonar transducer head. The collision avoidance sonar (usually the ST725) is directly controlled by the execution level for reliability and rapid response.

Path Planning and Replanning. Path planning is a tactical function. The strategic level contains the commanding officer (CO) and controls the overall mission plan. The CO decides (in general terms) where the ship will operate. Meanwhile, achieving the ordered track is the responsibility of the tactical level Officer of the Deck (OOD). To determine a safe route to the location the CO has requested, the OOD tells the tactical-level replanning department the desired location and the ship's present position. The sonar department (via the OOD) provides the replanning department with the current physical environment, i.e. where all the "circled" obstacles are. The replanning department takes this data and provides the OOD with the best path to the CO's ordered location after adding a safety distance around any obstacles. If a new obstacle is found by sonar while the ship is transiting, the OOD will call upon the replanning department to check the path. Replanning does not constantly process data but rather is called when the OOD needs it.

As a final step, smooth motion planning algorithms are applied to the output of the circle world path replanner in order to provide precise control of *Phoenix* and allow for rapid travel around obstacles without slowing into hover mode (Brutzman 92c) (Kanayama 95) (Leonhardt 96) (Davis 96a). Hover mode is inefficient when transiting waypoints, since it requires *Phoenix* to stop and maintain posture at a given location. Given the turning radius of a vehicle, smooth motion planning allows the vehicle to go from one point to another along a path that does not require the vehicle to perform instantaneous changes in direction. Thus the vehicle does not need to rotate in place when negotiating around obstacles. Replanner details are in (Leonhardt 96). Figure 9 illustrates the end-to-end process of detecting, classifying, localizing and avoiding a sonar obstacle.



- active sonar range/bearing returns
- line fits using parametric regression
- build polygon and classify obstacle
- safe standoff circle around polygon
- replan path around circled obstacles
- superimpose smooth path planning

Figure 9. Obstacle detection, classification, localization and avoidance.

7 STRATEGIC LEVEL

Prolog. The RBM strategic level is typically written in Prolog, a language for predicate logic. The strategic level implements a planning capability by sequencing mission phases and backtracking when necessary to provide appropriate guidance to the tactical level as portions of the mission succeed or fail. Strategic level design criteria follow in Figure 10.

- Symbolic computation only, contains mission-independent doctrine predicates and current mission guidance predicates
- No storage of internal vehicle or external world state variables
- Rule-based implementation, incorporating rule set, inference engine and working memory (if required)
- Non-interruptible, not event driven
- Directs tactical level via asynchronous message passing
- Messages may be either commands or queries requiring Boolean responses
- Operates in discrete (Boolean) domain independently of clock time
- Building blocks: goals
- Abstraction mechanism: goal decomposition (backwards chaining) and rule partitioning (forward chaining); both are based on goal-driven reasoning

Figure 10. RBM characteristics for strategic level (Byrnes 96).

Manually produced early versions of the strategic level worked properly but became large and complex. Strategic level code was streamlined by separating mission-independent doctrine from mission-specific guidance. With practice the strategic level Prolog code is relatively simple to read, produce and run. An example strategic level mission follows in Figure 11, where TASK might be a combination of GPS fix, drop marker, radio report, return home, etc.

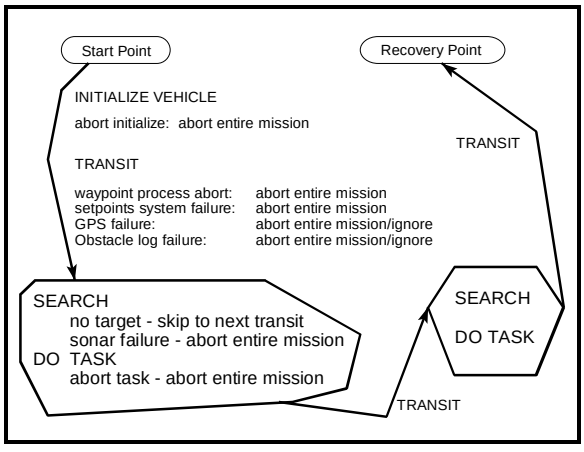


Figure 11. Strategic level representation of minefield search mission (Holden 95).

Mission Generation Expert System. The strategic level can also take the form of a deterministic finite automata (DFA). A mission controller initiates the phase associated with the current DFA node upon arrival, transitioning to a new node when the current node's phase completes successfully (or aborts because of a time out). A representative mission phase template appears in Figure 12. Individual tactic predecessors and successors can be composed using this template to create missions of arbitrary complexity (Davis 96a, 96b).

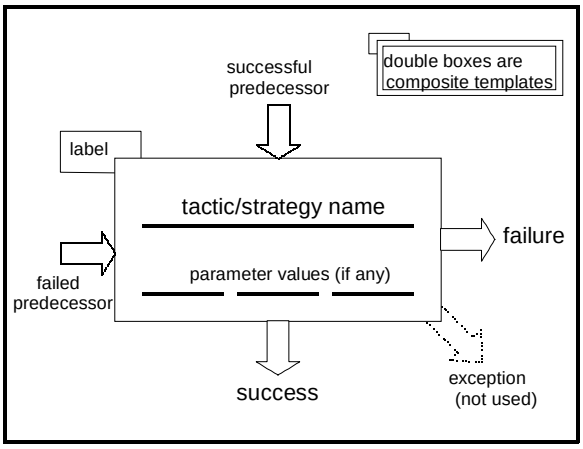


Figure 12. Template for tactic and strategy composition.

Advantages of the strategic level DFA structure are twofold. First, an arbitrary mission can be modeled simply as a set of phases that are executed in an order defined by the transitions of the DFA. Second, mission control using the Prolog search engine is powerful enough that complex behavior can be implemented without needing computationally intensive mathematical calculations. Arithmetic is confined to the tactical level, conceptual mission planning is confined to the strategic level.

Since a prime motivation for *Phoenix* is shallow water counter-mine operations, the mission generation process must be substantially simpler than writing Prolog programs if typical human operators are to deploy the AUV. One solution to this problem combines a graphical user interface for mission planning and specification together with a goal-driven expert system for strategic level code generation.

There are three aspects to the AUV Mission Generation Expert System. The first is a mission planning tool, which specifies vehicle launch and recovery positions and what the mission is supposed to accomplish. Means-ends analysis then computes a sequence of phases which can accomplish the desired mission. Failure of any single phase will cause a mission to either abort or follow an alternate failure-recovery phase (Byrnes 96). Since there may be multiple phase sequence solutions for a mission, each solution generated by the system is the next solution found as opposed to an optimal solution. In addition, missions generated through means-ends analysis are linear and proceed phase-by-phase to the end. In any case, users are allowed to choose among the candidate solutions generated (Davis 96a, 96b).

More complicated missions can take full advantage of this strategic level DFA structure. They are specified phase-by-phase using the second piece of the Mission Generation Expert System, the mission specification tool. This tool allows an experienced user who understands the DFA structure of the strategic level to define missions one phase at a time. Regardless of whether the mission planning tool or the mission specification tool is used, the system automatically checks input for correctness and logic and will not allow specification of an invalid mission (Leonhardt 96) (Holden 95) (Davis 96a, 96b).

The final aspect of this system is the code generation facility. By using specified phases, either the mission planning or mission specification tool, and templates for valid phase types (e.g. hover, search etc.) the system can generate executable code in either Prolog or C++. Earlier theses demonstrated that the strategic level can be equivalently instantiated using either the Prolog backwards chaining engine or the CLIPS forward chaining engine. Alternate languages are possible because there are multiple ways to plan. Backwards chaining can be unambiguously implemented using forward chaining, forward chaining can be unambiguously implemented using backwards chaining, and both can be implemented using fully enumerated decision graphs. Use of C++ has become possible because improved understanding and tighter constraints on mission primitives has eliminated the need for the full functionality of the Prolog search engine. Nevertheless such simplifications were only possible following extended experimentation using Prolog code.

Extensive testing of autogenerated Prolog and C++ code has been conducted in the virtual world, and successful in-water testing has been conducted at the *Phoenix* AUV test tank, Moss Landing Harbor and the NPS swimming pool. Further in-water tests are planned. Accomplishing our goal of simplifying mission generation is indicated by a significant reduction in the time required for mission coding (minutes when using the expert system as opposed to hours without it). Finally, syntactic programming errors have been completely eliminated by the source code autogeneration system and logical programming errors have been substantially reduced.

8 ROBOT NETWORKING

Perhaps surprisingly for a small robot, networking is a major consideration. Within the *Phoenix* AUV is an Internet-connectable local-area network (LAN). This enables network communications between and within the three software levels, external connectivity in laboratory via tether cable, and (optionally) external connectivity during harbor testing. Remote connection of the LAN to the campus Internet backbone is achieved using multiple wireless bridge boxes. Multicast Backbone (MBone) connectivity permits local or world-wide transmission of audio, video and DIS streams (Macedonia 94). World Wide Web links to online software documentation, multiple research group accounts and properly networked LANs with group access around campus further strengthened this software development collaboration. Ease of use and remote access translate into significant productivity gains and regular discovery of new capabilities. We expect to someday extend this approach underwater by developing Internet Protocol over Sea Water (IP/SW) connectivity (Brutzman 95a). Other network considerations are elaborated in Section 11 as part of virtual world connectivity.

9 VIRTUAL WORLD

The harsh environment in which an AUV must operate calls for extra precautions in its design to prevent damage to or loss of the vehicle. We have developed a medium-scale virtual environment which enables meaningful end-to-end testing of robot software and hardware in the laboratory (Figure 13). As noted in earlier work on the virtual world:

> It is tremendously difficult to observe, communicate with and test underwater robots, because they operate in a remote and hazardous environment where physical dynamics and sensing modalities are counterintuitive. An underwater virtual world can comprehensively model all necessary functional characteristics of the real world in real time. This virtual world is designed from the perspective of the robot, enabling realistic AUV evaluation and testing in the laboratory. 3D real-time graphics are our window into the virtual world, enabling multiple observers to visualize complex interactions. A networked architecture enables multiple world components to operate collectively in real time, and also permits world-wide observation and collaboration with other scientists interested in the robot and virtual world. (Brutzman 94)

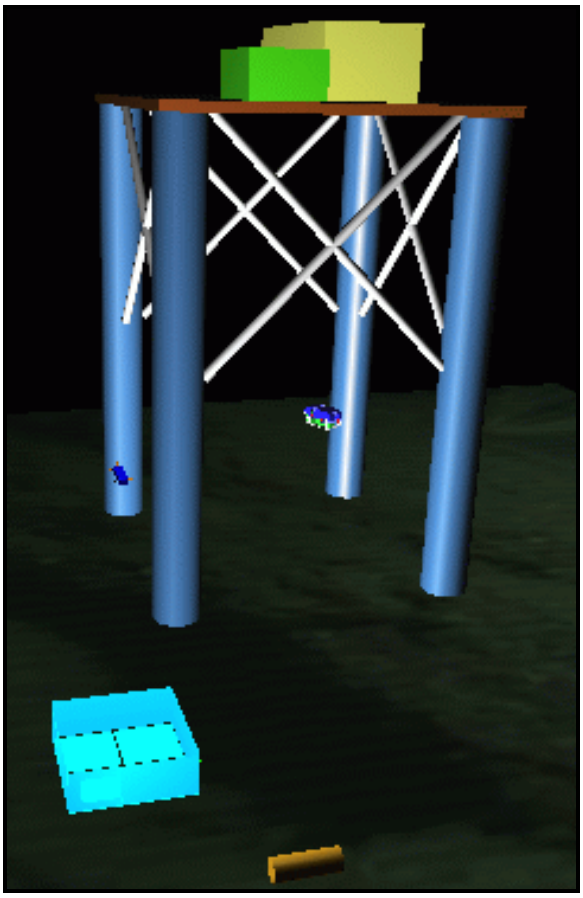


Figure 13. Underwater virtual world for an AUV (Brutzman 94).

The objective of the underwater virtual world is to reproduce real-world robot behavior with complete fidelity in the laboratory. Many questions pertain. What is the software architecture required to build an underwater virtual world for an autonomous underwater vehicle? How can an underwater robot be connected to a virtual world so seamlessly that operation in the real world or a virtual world is transparent to the robot? How can 3D real-time interactive computer graphics support wide-scale general access to virtual worlds? Specifically, how can computer graphics be used to build windows into an underwater virtual world that are responsive, accurate, distributable, represent objects in openly standardized formats, and provide portability to multiple computer architectures? Overview answers to these questions are provided here. Detailed analyses and example solutions are presented in (Brutzman 94). In effect, the virtual world requires a separate software architecture for networked world models that complements the robot software architecture.

The real world is a big place. Virtual worlds must similarly be comprehensive and diverse if they are to

permit credible reproductions of real-world behavior. A variety of software components have been shown necessary. In every case, 3D real-time visualization has been a crucial tool in developing AUV software. Ways to scale up and arbitrarily extend the underwater virtual world to include very large numbers of users, models and information resources are also incorporated in this work.

Virtual world capabilities were utilized for testing and verification throughout the software development process. Use of this tool allows a number of programmers to work independently and in concert. Virtual world capabilities have been incrementally improved to match increased vehicle software capabilities, such as hydrodynamics and controller response rendering (Figure 14). Scientific visualization techniques have provided further significant benefits (Brutzman 95b).

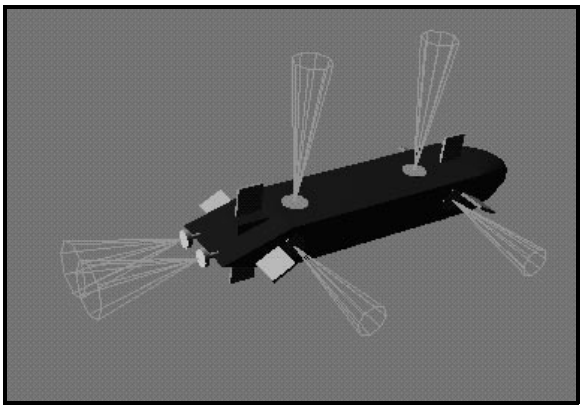


Figure 14. Detailed hydrodynamics and control visualization is essential.

10 VISUALIZING CONTROL ALGORITHMS

Designing an AUV is complex. Many capabilities are required for an underwater mobile robot to act capably and independently. Stable physical control, motion control, sensing, path planning, mission planning, replanning and failure recovery are example software components that must be solved individually for tractability. The diversity and dissimilarity of these many component subproblems precludes use of a single monolithic solution paradigm.

Vehicle control algorithms are implemented using either thrusters (hovering modes), planes/rudders/ propellers (cruise modes) or all effectors in combination. Control algorithms for the following behaviors are included: depth control, heading control, open-loop rotation, open-loop lateral motion, waypoint following and hovering. Control algorithms are permitted to operate both thrusters and planes/rudders/propellers simultaneously when such operation does not provoke mutual interference. Most *Phoenix* control code has been developed and tested in conjunction with the construction of a real-time six degree-of-freedom hydrodynamics model. Design, tuning and optimization of control algorithms in isolation and in concert is the subject of active research (Healey 93, 96) (Fossen 94) (Marco 96a) and remains an important area for future work. Control algorithm robustness is a particularly important topic since potentially fatal nonlinear instabilities are possible and vehicle reliability is paramount.

Typical efforts at hydrodynamic development are based on mental interpretation of multiple time-series such as Figure 15. Dozens of two-dimensional time-series plots are necessary for quantitative performance analysis, but this approach remains notoriously difficult to use when attempting to mentally integrate and visualize all aspects of vehicle behavior. The successes of individual control algorithms created as part of this effort were highly dependent on 2D and 3D visualization techniques. Complete derivations of the full hydrodynamics model and corresponding control equations are in (Brutzman 94, 96c).

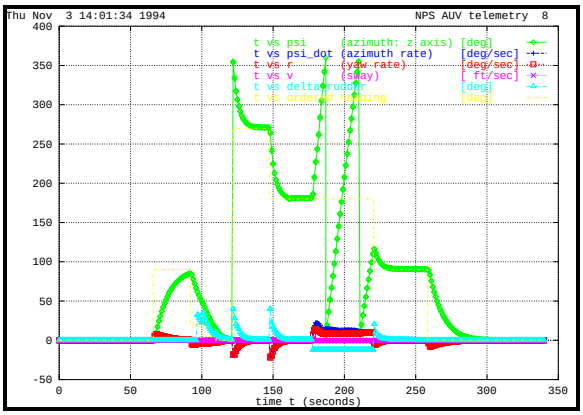


Figure 15. Representative time-series behavior plot.

An example challenging scenario for an AUV is evaluating vehicle control stability when transitioning from stable submerged control to intentional surface broaching in Figures 16 and 17. This scenario exercises the real-time buoyancy model developed in (Bacon 95). Real-time 3D observation of such scenes is an essential tool when developing and testing algorithmic models.

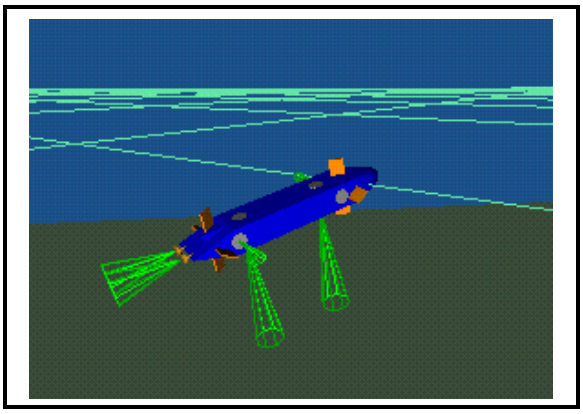


Figure 16. Evaluating control response while broaching.

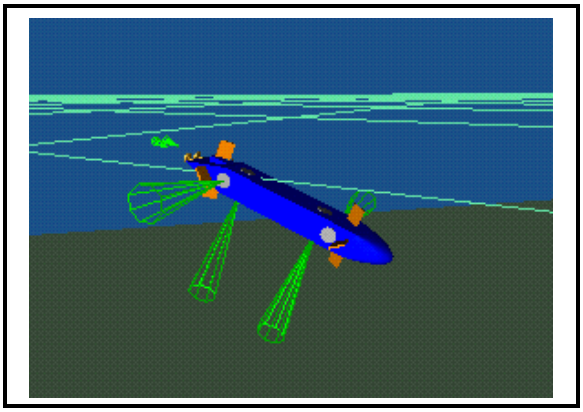


Figure 17. Evaluating control response after broaching.

11 AUV-VIRTUAL WORLD COMMUNICATIONS

Since RBM is a multilevel architecture, communications between levels must be formally defined. Communications between robot and virtual world must also be clearly specified. Defining communications includes establishing a physical path for data transfer as well as defining the syntax and protocol of exchanged messages. Our design objectives include reliability and clarity so that messages are easily created and easily understood, either by software processes or by people. Details follow in order to illustrate the precise relationships between robot, virtual world and graphics-based user viewing windows.

Two kinds of messages are defined for use between robot and virtual world. The first is the telemetry vector, which is a list of all vehicle state variables pertinent to hydrodynamic and sensor control. Telemetry vectors are passed as a string type. The second kind of messages allowed are free-format commands. Free-format command messages are also string types, starting with a predefined keyword and followed by entries which may optionally have significance depending on the initial keyword. Messages with unrecognized keywords are treated as comments. These two kinds of messages (telemetry and commands) can be used for any communication necessary among robot-related entities. Employment of string types facilitates data transfer between different architectures, data transfer via network sockets, and file storage. String types also ensure that all communications are readable by both robot and human, a trait that is particularly useful during debugging. An open format for command messages permits any user or new application to communicate with little difficulty.

Within the AUV, the basic communications flow between execution level and tactical level is straightforward. All telemetry vectors are sent from the execution level to the tactical level, providing a steady stream of time-sensitive, rapidly updated information. The tactical level may send commands to the execution level as desired, and the execution level may return informational messages between telemetry vectors as appropriate. Nonadaptive tactical level functionality can also be provided by carrying out prescripted mission command files. Telemetry vector records and command messages are logged in separate mission output files for post-mission analysis and replay.

The telemetry vector serves several essential purposes. In addition to providing a steady stream of information from the execution level to the tactical level, the telemetry vector also serves as the data transfer mechanism between execution level and virtual world. Efficient communications between robot and virtual world are essential if rapid real-time 10 Hz robot response is to be maintained. The telemetry record is a concise and complete way to support all of these data communications requirements. Figure 18 shows in detail how the flow of control proceeds and the telemetry vector is modified during each *sense-decide-act* cycle.

Robot execution software is designed to operate both in the virtual world and in the real world. While sensing in the virtual world, distributed hydrodynamics and sonar models fill in pertinent telemetry vector slots. While sensing in the real world, actual sensors and their corresponding interfaces fill in pertinent telemetry vector slots. In either case, the remainder of the robot execution program which deals with tactical communications, command parsing, dynamic control, interpretation etc. is unaffected. While operating in the virtual world, robot propulsion and sensor commands are communicated via the same telemetry vector. While operating in the real world, robot

propulsion and sensor commands are sent directly to hardware interfaces for propellers, thrusters, planes, rudders, sonar steering motors, etc. Again almost all parts of the robot execution program are completely unaffected by this difference. This networked architecture is essentially transparent to the robot, permitting identical AUV operation in the real world or virtual world.

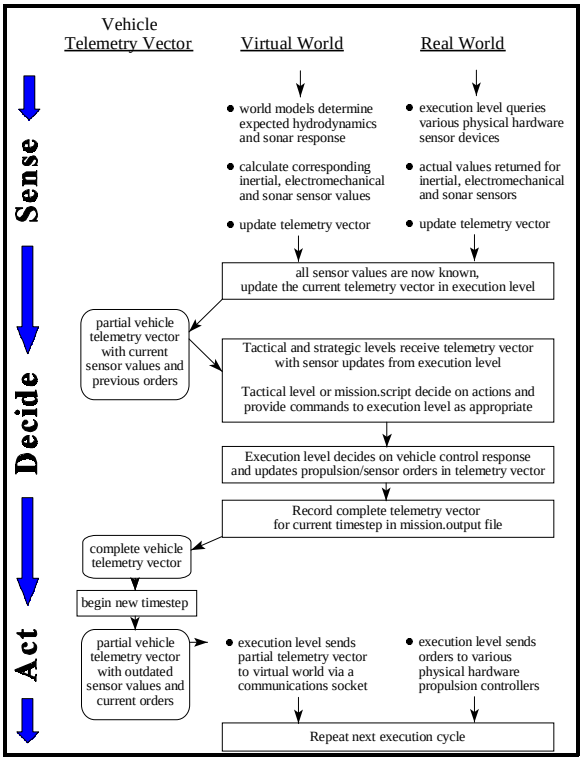


Figure 18. Data flow via the telemetry vector during each *sense-decide-act* cycle.

The telemetry vector is therefore a key data transfer mechanism. Telemetry vector updates also define the communication protocol between execution level and virtual world. As might be expected, this works well because the execution level program follows the common robotics cyclic paradigm of *sense-decide-act*. Figure 19 provides an overview of the telemetry vector update sequence as an alternate means of portraying the validity of this approach. Given the perhaps-worst-case computational complexity of underwater world models, this networked virtual world software architecture for real-time performance in the laboratory also appears applicable to other robot domains.

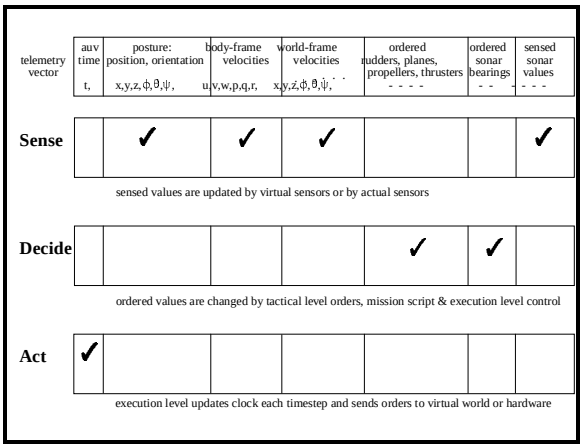


Figure 19. Telemetry vector modifications during each *sense-decide-act* cycle.

12 INTERACTIVE 3D GRAPHICS

Several important requirements are needed for the creation of object-oriented graphics viewers for visualizing a large-scale virtual world. Open standards, portability and versatility are emphasized over platform-specific performance considerations in order to support scaling up to very large numbers of users, platform types and information sources. The *OpenInventor* graphics toolkit and scene description language has all of the functionality needed. The potential integration of network connections to logically extend graphics programs is also examined. Open standards, portability and versatility are emphasized over platform-specific performance considerations in order to support scaling up to very large numbers of users, platform types and information sources.

A good graphics toolkit for building a virtual world viewer has many requirements to fill (Foley, van Dam 90). Rendered scenes need to be realistic, rapidly rendered, permit user interaction, and capable of running on both low end and high end workstations. Graphics programmers must have a wide range of tools to permit interactive experimentation and scientific visualization of real-world datasets (Thalmann 90). The ability to read multiple data formats is also important when using scientific and oceanographic datasets. Scientific data format compatibility can be provided by a number of data function libraries which are open, portable, reasonably well standardized and usually independent of graphics tools (Fortner 92). Viewer programs need to be capable of examining high-bandwidth information streams and large archived scientific databases. Thus the ability to preprocess massive datasets into useful, storable, retrievable graphics objects will be particularly important as we attempt to

scale up to meet the sophistication and detail of the real world. Adequate standardization of computer graphics and portability across other platforms is also desirable but has been historically elusive.

OpenInventor is an object-oriented 3D graphics toolkit for graphics applications design (Strauss 92). Based on the *Open GL* graphics library, *OpenInventor* provides high-level extensions to the ***C***++ (or ***C***) programming language and a scene description language. It is designed to permit graphics programmers to focus on what to draw rather that how to draw it, creating scene objects that are collected in a scene database for viewpoint-independent rendering.

The ability to store graphics objects as readable, editable files is especially appealing for the creation of large-scale virtual worlds. Since the performance of computer graphics is highly dependent on the computational complexity of scenes to be rendered, it is inevitable that truly large-scale world scene databases will eventually overload viewing graphics workstations. Such overload will occur regardless of the efficiency of viewpoint culling algorithms and graphics pipeline optimizations, unless partitionable and networked scene databases are used. Furthermore, since populating a virtual world is a task that needs to be open and accessible to large numbers of people, an open graphics data standard is needed for virtual world construction. The ability to selectively load graphics objects and scenes from files is an important distribution mechanism which can take advantage of Web connectivity.

Ubiquitous portability for analytic, hypermedia, network, multicast and graphics tools is therefore an essential feature for virtual world model builders. A superior alternative is now available using the Virtual Reality Modeling Language (VRML) specification (Carey 96). VRML is the Web standard for interactive 3D representation. VRML scene description files are the best approach for object definitions in a large-scale virtual world (Brutzman 96d).

13 SONAR VISUALIZATION

Sensor differences distinguish underwater robots from ground, air and space-based robots. Since the oceans are generally opaque to visible light at moderate-to-long ranges, vision-based video systems are ordinarily of use only at short distances and are unreliable in turbid water. Vision systems also usually require intense light sources which deplete precious energy reserves. In comparison to underwater computer vision, active and passive sonar (acoustic detection) has long been a preferred sensing method due to the long propagation ranges of sound waves underwater.

However, sound waves can be bent by variations in depth, temperature and salinity. A variety of problems including ambient noise, multipath arrival, fading, shadow layers, masking and other effects can make sonar use difficult. Since active sonar typically provides good range values with approximate bearing values, algorithms for sonar recognition are much different than vision algorithms. In the short sonar ranges used by *Phoenix*, simple error probabilities and linear geometric sonar relationships are adequate. Figure 20 shows the perspective gained by observing AUV sonar from an "over the shoulder" perspective, one of several vantage points needed when developing sonar classification algorithms.

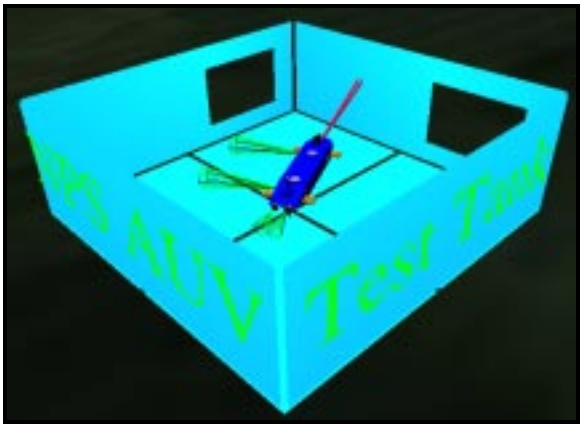


Figure 20. Local viewpoint of active sonar in test tank.

Since sonar is the most effective detection sensor used by underwater vehicles, sonar visualization is particularly important when designing and evaluating robot software. Sonar parameters pertinent to visualization and rendering include sound speed profile (SSP), highly-variable sound wave path propagation, and sound pressure level (SPL) attenuation. Several questions are prominent. How can a general sonar model be networked to provide real-time response despite high computational complexity? How can scientific visualization techniques be applied to outputs of the sonar model to render numerous interacting physical effects varying in three spatial dimensions and time? Initial investigations indicate that this area may yield significant results. The high dimensionality of sonar data is best served by scientific visualization techniques.

Sonar sensing is crucially important (Stewart 92). Previously only a single geometric sonar model was available for *Phoenix*, derived by hand to model the AUV test tank (Figure 21). Although effective in a small regular volume, this approach was too limited and did not permit easy addition of artificial targets or

obstacles. We adapted the computational geometry routines included in the *OpenInventor* interactive 3D graphics library to shoot rays into the scene database to produce a general geometric sonar model. Now the same scene database (made up of *OpenInventor* and VRML files) can be used for both virtual world visualization and real-time 3D sonar ray intersection calculations (Figure 22) (Davis 96a) (Brutzman 96b).

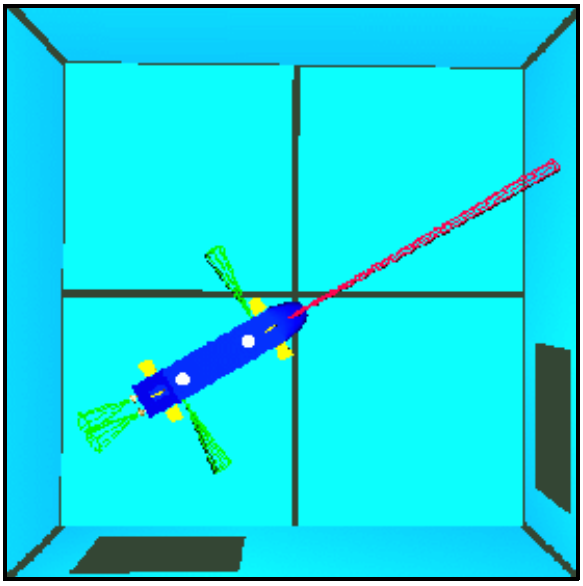


Figure 21. Manually derived geometric sonar model for AUV test tank (Brutzman 94).

14 EXPERIMENTAL TEST RESULTS

Once *Phoenix* functionality was correct in the virtual world, test tank experiments were conducted to fine tune hardware and properly move the AUV through the water. Diving, forward, backward, lateral and rotational movement checks were all performed during these test tank experiments. However, the calibration of speeds during these movements could not be tested due to the relatively small size of the test tank (6m x 6m x 2m deep).

The next vehicle tests were performed in the relatively calm sea water harbor in Moss Landing California. A variety of logistical problems were overcome but a seemingly endless series of minor hardware failures then thwarted each attempt to run a complete minefield search. Although a complete mission was never accomplished beginning to end, all components of the mission were individually exercised. We now believe that the functionality and logic of the AUV software is correct (Brutzman 96b). Remaining tests include repeated mission testing, verification of aggregate software behavior under a variety of scenarios, tuning of control constants, and validation of both hydrodynamics and sonar models in the virtual world. Recent results include precise vehicle maneuvering and rendezvous with a docking tube (Davis 96a, 96b) (Figure 23). Much more experimental testing awaits.

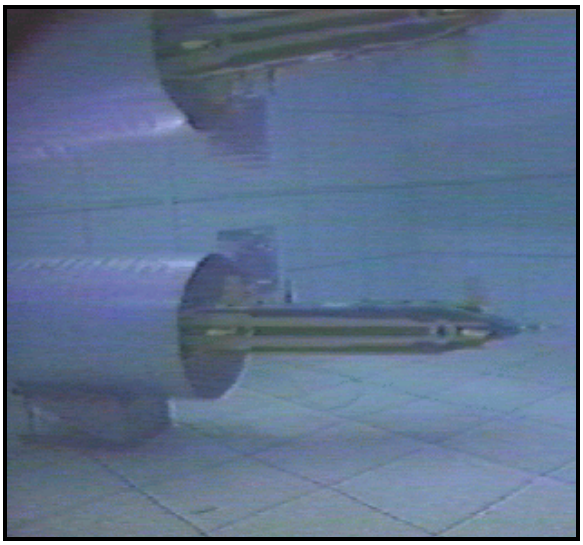


Figure 22. *Phoenix* AUV maneuvering to enter a docking tube using onboard sonar (Davis 96a, 96b).

15 FUTURE WORK

An underwater vehicle which can transit through waypoints and hover in the presence of currents enables a variety of capabilities which are not possible for vehicles that must retain forward way to remain hydrodynamically stable. We intend to examine whether the *Phoenix* hull form can stably approach and neutralize a moored mine-like object. Figure 24 is a notional diagram that shows how sonar can be used to carefully approach a target broadside, keep station against the ocean current, take confirming video, and attach a beacon or neutralizing device using a simple one- or two-degree-of-freedom effector. For low sea states, we see few limiting factors in this approach.

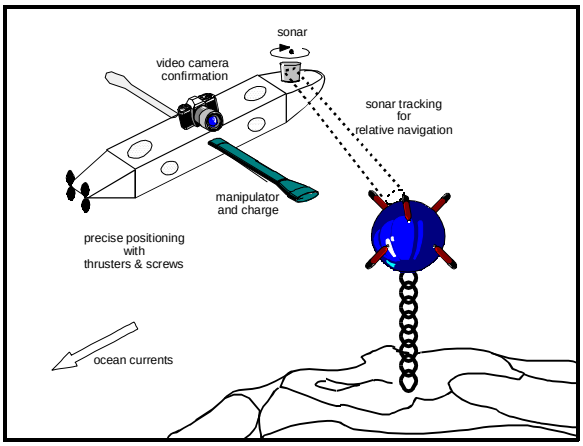


Figure 23. A mobile stable AUV might precisely place an explosive charge on an underwater mine.

Phoenix is only directly controllable in five degrees of freedom since roll is unconstrained. Pitch stabilization is straightforward using vertical thrusters. Testing will determine whether roll stabilization is also necessary, perhaps by using an additional thruster. We are further interested in development of automatic diagnostics that reconfigure control algorithms to handle equipment faults. We also intend to explore local measurement of cross-body ocean current flow using acoustic doppler current profilers (ADCPs), in order to permit precise maneuvering in the midst of highly varying flow fields and high sea states. Finally, future work on underwater virtual world networked graphics includes compatibility with common Web browsers using the Virtual Reality Modeling Language (VRML) (Brutzman 96d).

16 CONCLUSIONS

The underwater environment is extremely challenging for robots. Counterintuitive hydrodynamics response, poor visual capabilities, complex sonar interactions, communications inaccessibility and power endurance are significant design constraints. Robot builders must provide stable control and reliable operation at all times due to the unacceptably high cost of failure. A variety of AI processes must be used for planning, sensing and other complex tasks.

Systems integration is significant due to the many sensors and effectors required for nontrivial operation. The *Phoenix* AUV demonstrates that a three-layer robot architecture can be effective at combined system control over time scales ranging from hard-real-time sense-decide-act response to temporally unconstrained mission planning.

Using an underwater virtual world for interactive 3D graphics rendering is an essential capability for effective AUV development. The networked software architecture and various results described here demonstrate that a real-time physically based underwater virtual world is feasible. It enables repeated testing of all aspects of underwater vehicle control, stability, sensing, autonomy and reliability. Graphics viewer requirements include scientific visualization and portability across multiple platforms. The use of multicast DIS messages, Web access and VRML scene descriptions that include dynamic behaviors promise the possibility of scaling to very large numbers of participants. Network connectivity allows us to use the global Internet as a direct extension of our desktop computers, permitting global collaboration on a routine basis.

After years of effort, the RBM architecture is fully instantiated onboard the *Phoenix* AUV and is being successfully tested and refined by in-water testing. A networked underwater virtual world has been crucial to this development project. Experimental results indicate we are close to demonstrating that affordable underwater robots can operate autonomously in challenging environments.

17 REFERENCES

Bachmann, E.R., McGhee, R.B., Whalen, R.H., Steven, R., Walker, R.G., Clynch, J.R., Healey, A.J. and Yun, X.P., "Evaluation of an Integrated GPS/INS System for Shallow-Water AUV Navigation (SANS)," *Proceedings of the IEEE Oceanic Engineering Society Conference AUV 96*, Monterey California, June 3-6 1996, pp. 268-275.

Bacon, Daniel Keith Jr., *Integration of a Submarine into NPSNET*, Master's Thesis, Naval Postgraduate School, Monterey California, September 1995. Available via *http://www-npsnet.cs.nps.navy.mil/npsnet*

Bottoms, Al, chair and editor, *Symposium on Autonomous Vehicles for Mine Countermeasure*s, Naval Postgraduate School, Monterey California, April 1995.

Brooks, Rodney A., "A Robust Layered Control System for a Mobile Robot," IEEE Journal of Robotics and Automation, vol. RA-2 no. 1, March 1986, pp. 14-23.

Brutzman, Donald P. and Compton, Mark A., "AUV Research at the Naval Postgraduate School," *Sea Technology*, vol. 32 no. 12, December 1991, pp. 35-40.

Brutzman, Donald P., "From virtual world to reality: designing an autonomous underwater robot," *American Association for Artificial Intelligence (AAAI) Fall Symposium on Applications of Artificial Intelligence to Real-World Autonomous Mobile Robots*, Cambridge Massachusetts, October 23-25 1992, pp. 18-22. Available at *ftp://taurus.cs.nps.navy.mil/pub/auv/aaai92ws.ps.Z*

Brutzman, Donald P., Compton, Mark A. and Kanayama, Yutaka, "Autonomous Sonar Classification using Expert Systems," *Proceedings of the IEEE Oceanic Engineering Society Conference OCEANS 92*, Newport Rhode Island, October 26-29 1992, pp. 554-559. Available at *ftp://taurus.cs.nps.navy.mil/pub/auv/oceans92.ps.Z*

Brutzman, Donald P., *NPS AUV Integrated Simulator*, Master's Thesis, Naval Postgraduate School, Monterey California, March 1992. Includes video appendix.

Brutzman, Donald P., "Beyond intelligent vacuum cleaners," *American Association for Artificial Intelligence (AAAI) Fall Symposium on Applications of Artificial Intelligence for Instantiating Real-World Agents*, Raleigh North Carolina, October 22-24 1993, pp. 23-25. Available at *ftp://taurus.cs.nps.navy.mil/pub/auv/aaai93ws.ps.Z*

Brutzman, Donald P., *A Virtual World for an Autonomous Underwater Vehicle*, Ph.D. Dissertation, Naval Postgraduate School, Monterey California, December 1994. Includes video appendix. Available at *http://www.stl.nps.navy.mil/~brutzman/dissertation*

Brutzman, Don and Reimers, Stephen, "Internet Protocol over Seawater (IP/SW): Towards Interoperable Underwater Networks," *Ninth International Symposium on Unmanned Untethered Submersible Technology (UUST) 95*, University of New Hampshire, Durham New Hampshire, September 25-27 1995. Available at *ftp://taurus.cs.nps.navy.mil/pub/auv/ipoversw.ps*

Brutzman, Don, "Virtual World Visualization for an Autonomous Underwater Vehicle," *Proceedings of the IEEE Oceanic Engineering Society Conference OCEANS 95*, San Diego California, October 12-15 1995, pp. 1592-1600. Available at *ftp://taurus.cs.nps.navy.mil/pub/auv/oceans95.ps.Z*

Brutzman, Don, "Tutorial: Virtual World for an Autonomous Underwater Vehicle (AUV)," *IEEE Oceanic Engineering Society Conference OCEANS 96*, Fort Lauderdale Florida, September 23-26 1996. Available at *http:/www.stl.nps.navy.mil/~auv/uvw_tutorial.html*

Brutzman, Don, Burns, Mike, Campbell, Mike, Davis, Duane, Healey, Tony, Holden, Mike, Leonhardt, Brad, Marco, Dave, McClarin, Dave, McGhee, Bob and Whalen, Russ, "NPS *Phoenix* AUV Software Integration and In-Water Testing," *Proceedings of the IEEE Oceanic Engineering Society Conference AUV 96*, Monterey California, June 3-6 1996, pp. 99-108. Available at *ftp://taurus.cs.nps.navy.mil/pub/auv/auv96.ps*

Brutzman, Don, "Graphics Internetworking: Bottlenecks and Breakthroughs," chapter, *Digital Illusion*, Clark Dodsworth editor, Addison-Wesley, Reading Massachusetts, to appear 1996. Available at *http:// www.stl.nps.navy.mil/~brutzman/breakthroughs.html*

Brutzman, Don, *NPS Phoenix AUV Software Reference*, November 1996. Available at *http:// www.stl.nps.navy.mil/~auv/software_reference.html*

Burns, Mike, *An Experimental Evaluation and Modification of Simulator-based Vehicle Control Software for the Phoenix Autonomous Underwater Vehicle (AUV)*, Master's Thesis, Naval Postgraduate School, Monterey California, April 1996. Available at *http://www.cs.nps.navy.mil/research/auv*

Byrnes, Ronald Benton Jr., *The Rational Behavior Model: A Multi-Paradigm, Tri-Level Software Architecture for the Control of Autonomous Vehicles*, Ph.D. Dissertation, Naval Postgraduate School, Monterey California, March 1993.

Byrnes, Ronald B., Healey, Anthony J., McGhee, Robert B., Nelson, Michael L., Kwak, Se-Hung and Brutzman, Donald P., "The Rational Behavior Software Architecture for Intelligent Ships," *Naval Engineers' Journal*, March 96, pp. 43-55.

Campbell, Michael Scott, *Real-Time Sonar Classification for Autonomous Underwater Vehicles*, Master's Thesis, Naval Postgraduate School, Monterey California, March 1996.

Carey, Rikk, Marrin, Chris and Bell, Gavin, "The Virtual Reality Modeling Language (VRML) Version 2.0 Specification," International Standards Organization/ International Electrotechnical Commission (ISO/IEC) draft standard 14772, August 4 1996. Available via the VRML Repository at *http://www.sdsc.edu/vrml*

Curtin, Thomas B., Bellingham, James G., Catipovic, Josko and Webb, Doug, "Autonomous oceanographic sampling networks," *Oceanography*, vol. 6, 1993, pp. 86-94. Additional information at *http:// web.mit.edu/afs/athena/org/s/seagrant/www/auv.htm*

Davis, Duane, *Precision Maneuvering and Control of the Phoenix Autonomous Underwater Vehicle for Entering a Recovery Tube*, Master's Thesis, Naval Postgraduate School, Monterey California, September 1996. Includes video appendix. Available via *http://www.cs.nps.navy.mil/research/auv*

Davis, D., Brutzman, D., Leonhardt, B., McGhee, R., "Operational Mission Planning and Mission Control for the *Phoenix* Autonomous Underwater Vehicle," *IEEE Journal of Oceanic Engineering*, in review, 1996.

Flagg, Marco, "Submersible Computer for Divers, Autonomous Applications," *Sea Technology*, vol. 35 no. 2, February 1994, pp. 33-37.

Foley, James D, van Dam, Andries, Feiner, Steven K. and Hughes, John F., *Computer Graphics: Principles and Practice*, second edition, Addison-Wesley, Reading Massachusetts, 1990.

Fortner, Brand, *The Data Handbook: A Guide to Understanding the Organization and Visualization of Technical Data*, Spyglass Inc., Champaign Illinois, 1992.

Fossen, Thor I., *Guidance and Control of Ocean Vehicles*, John Wiley & Sons, Chichester England, 1994.

Healey, A.J. and Lienard, D., "Multivariable Sliding Mode Control for Autonomous Diving and Steering of Unmanned Underwater Vehicles," *IEEE Journal of Oceanic Engineering*, vol. 18 no. 3, July 1993, pp. 327-339.

Healey, A.J., Marco, D.B., McGhee, R.B., Brutzman, D.P. and Cristi, R., "Evaluation of the NPS Phoenix Autonomous Underwater Vehicle Hybrid Control System," *Proceedings of the American Controls Conference (ACC) 95*, San Francisco California, June 1995.

Healey, A. J., Marco, R.B. and McGhee, R.B., "Autonomous Underwater Vehicle Control Coordination Using a Tri-Level Hybrid Software Architecture," *Proceedings of the IEEE Robotics and Automation Conference*, Minneapolis Minnesota, April 1996.

Holden, Michael J., *Ada Implementation of Concurrent Execution for Multiple Tasks in the Strategic and Tactical Levels of the Rational Behavior Model for the NPS AUV*, Master's Thesis, Naval Postgraduate School, Monterey California, September 1995.

IEEE Standard for Distributed Interactive Simulation (DIS) -- Communication Service and Profiles, IEEE Standard P1278.1, Institute of Electrical and Electronic Engineers, New York, 1995. Information available at *http://www.sc.ist.ucf.edu/~STDS*

Kanayama, Yutaka, "Introduction to Motion Planning," CS4313 Lecture Notes, Naval Postgraduate School, Monterey California, March 1995.

Leonhardt, Bradley J., *Mission Planning and Mission Control Software for the Phoenix Autonomous Underwater Vehicle (AUV): Implementation and Experimental Study*, Master's Thesis, Naval Postgraduate School, Monterey California, March 1996. Available at *http://www.cs.nps.navy.mil/research/auv*

Macedonia, Michael R. and Brutzman, Donald P., "MBone Provides Audio and Video Across the Internet," IEEE *COMPUTER*, vol. 27 no. 4, April 1994, pp. 30-36. Available at *ftp://taurus.cs.nps.navy.mil/pub/i3la/mbone.html*

Marco, D. B. and Healey, A. J., "Local-Area Navigation Using Sonar Feature Extraction and Model-Based Predictive Control," *IEEE Symposium on Autonomous Underwater Vehicle Technology*, Monterey California, June 3-6 1996, pp. 67-77.

Marco, D. B., Healey, A. J. and McGhee, R.B., "Autonomous Underwater Vehicles: Hybrid Control of Mission and Motion," *Autonomous Robots*, vol. 3, 1996, pp. 169-186.

McClarin, David W., *Discrete Multi-Mode Kalman Filtering of Navigation Data for the Phoenix Autonomous Underwater Vehicle*, Master's Thesis, Naval Postgraduate School, Monterey California, March 1996.

Moravec, Hans, "The Stanford Cart and the CMU Rover," *Proceedings of the IEEE*, vol. 71 no. 7, July 1983, pp. 872-884.

Sayers, Craig P., Yoerger, Dana R., Paul, Richard P. and Lisiewicz, John S., "A Manipulator Work Package for Teleoperation from Unmanned Untethered Vehicles - Current Feasibility and Future Applications," *International Advanced Robotics Programme (IARP) on Subsea Robotics*, Toulon France, March 27-29 1996. Additional information at *http://www.dsl.whoi.edu*

Shank, Roger C., "Where's the AI?," *AI Magazine*, vol. 12 no. 4, Winter 1991, pp. 38-49.

Smith, Samuel M. and Dunn, Stanley E., "The Ocean Voyager II: An AUV Designed for Coastal Oceanography," *Proceedings of the IEEE Oceanic Engineering Society Conference Autonomous Underwater Vehicles (AUV) 94*, Cambridge Massachusetts, July 19-20 1994, pp. 139-147. Additional information available at *http://www.oe.fau.edu/AMS*

Stevens, Richard W., *Advanced Programming in the Unix Environment*, Addison-Wesley, Reading Massachusetts, 1992.

Stewart, W. Kenneth, "Visualization resources and strategies for remote subsea exploration," *The Visual Computer*, Springer-Verlag, vol. 8 no. 5-6, June 1992, pp. 361-379.

Strauss, Paul S. and Carey, Rikk, "An Object-Oriented 3D Graphics Toolkit," *COMPUTER GRAPHICS*, vol. 26 no. 2, July 1992, pp. 341-349.

Thalmann, Daniel, editor, *Scientific Visualization and Graphics Simulation*, John Wiley & Sons, Chichester Great Britain, 1990.

Torsiello, Kevin, *Acoustic Positioning of the NPS Autonomous Underwater Vehicle (AUV II) During Hover Conditions*, Engineer's Thesis, Naval Postgraduate School, Monterey California, March 1994.

Yuh, Junku, editor, *Underwater Robotic Vehicles: Design and Control*, TSI Press, Albuquerque New Mexico, 1995.

18 SOFTWARE AND DOCUMENTATION

All source code, support files and compiled executable programs are available via the Internet (Brutzman 96a). This software reference includes help files, *Phoenix* software, 3D graphics viewer, hydrodynamics, sonar modeling, networking and Multicast Backbone (MBone) resources. AUV dynamics software is parameterizable for other vehicles and all work is in the public domain. Available at *http://www.stl.nps.navy.mil/~auv*

Acknowledgements. The authors thank Mike Zyda and Yutaka Kanayama for help and advice during the conduct of this research. We are also grateful to approximately eight dozen colleagues and students of the NPS Center for AUV Research who have made valuable contributions to *Phoenix*. Financial support for this ongoing work has been provided by the National Science Foundation under Grant BCS-9306252 and the Naval Postgraduate School Research Initiation Program.

To appear: *AI-Based Mobile Robots*, Kortenkamp, David, Bonasso, Peter and Murphy, Robin, editors, MIT/AAAI Press, Cambridge Massachusetts, 1997.

This chapter is available online at *http://www.stl.nps.navy.mil/~auv/aimr.html* and *http://www.stl.nps.navy.mil/~auv/aimr.ps*